Competitive Analysis

Comparing the e-TEST suite and Mercury Astra

Published on December 21, 2000
Version 2.0

INTRODUCTION

This document presents the main differences between the RSW e-TEST suite and Mercury Astra, for functional and load testing of Web-based applications. It details the main advantages and drawbacks of the two products.

RECORDING AND SCRIPTING TECHNOLOGIES

RECORDING TECHNOLOGY AND TEST CASES

Both products have technology to capture the interactions of the user with the Web application.

Astra uses two technologies:

- A browser-level recorder that starts an instance of a browser and lets the user navigate through the application. This recorder creates what is called browser-based scripts.
- A HTTP recorder that captures the low-level HTTP calls to the Web server. This recorder creates HTTP-based scripts.

The problem is that **once you have two scripts for the same test, these two scripts need to be maintained in parallel,** as there is no correlation between them. The browser-based script will be used for functional testing, and low-volume load testing. But since it is extremely resource-intensive**, it cannot be used for large-volume load testing** (see "Scalability" below).

e-Tester uses an advanced COM technology to instantiate the IE browser installed on the PC, and record through this browser. There is no need to call an external browser since the browser is being seamlessly integrated into e-Tester.

The major difference resides in the capture technology. e-Tester captures the entire Document Object Model of each page the user navigates to or that the browser loads automatically, thus creating **all the test cases** that will be used for functional and load testing. Capturing the DOM **is the only way to get an exhaustive view of each page of the application** – however it does not prevent cross-browser testing and e-Tester also supports Netscape (rather than using the least common denominator, e-Tester also builds test cases specifically for the Netscape Document Object Model). For highly dynamic Web sites, in which content changes often, e-Tester will let the user selectively turn off certain test cases that would fail even though the application is working.

Astra **does not record the content of each page**, and instead records the request. This means that Astra does not generate test cases for each element of the page – links, forms, images, even the HTML itself. The Astra user needs to manually add "Checkpoints", or test cases that verify that a certain text is present on a page, or counts the links or images, etc. Unlike the e-Tester script validation, a test built with Astra does not check all the specific elements of the Document Object Model – and the more manual work there is to do to create an exhaustive script, **the more chances there are that some elements will be missed**.

EVENTS CAPTURE

As mentioned above, Astra features two different recording modes, which produce two different scripts. The browser-based script contains all the browser events, but the HTTP-based script only contains calls to the Web server and as such **is unable to reproduce client-side activity**, such as calls to JavaScript functions.

e-Tester creates only one Visual Script, that is used both for functional testing and load testing. There is no need to have two scripts – one that scales and one that does not. This script implements – without any need to

write code – all client-side activity and results in transactions being sent to the server. **The Visual Script can easily scale 1,000 virtual users or more on a single PC** (see "Scalability" below).

When dynamic session information is used, again Astra works fine in browser mode, but the HTTP-based script requires manual modifications in order to be able to handle the application at all. If the user does not go through complex modifications to the generated code to manually extract the session identifiers to variables and then replace the variables in each subsequent call, the playback continues to use the originally recorded values – resulting in playback failure. **With Astra, the only way to playback a script without editing code beforehand, is to use the browser-based script that does not scale** (see "Scalability" below).

On the other hand**, e-Tester and e-Load implement a breakthrough technology that automatically handles these sessions identifiers and other technical information** generated by the Web application, making the Visual Script play successfully every time, **without a need to edit any script code**. A script recorded with e-Tester can be played back immediately in e-Load, with 1,000 virtual users or more on a single PC.

SCRIPTING TECHNOLOGY

In the browser-based script, Astra implements a dual view of the actions recorded in the Web application – a tree view and a VBScript view. At first glance, this may appear to offer the user additional flexibility. In reality, the VBScript script in Astra is **only a series of instructions** telling the program which page to navigate, and are rather like high-level function calls. There is absolutely no control over the fine execution of the functions. In addition, VBScript is not a real development environment (**you don't even get a debugger with Astra's VBScript**, you need to install it separately!).

Astra's HTTP-based scripts (the ones that scale for load testing) are made of calls to HTTP functions and are **written in a proprietary language, extremely complex to learn**. This is in fact the same programming language as the one used by LoadRunner, Mercury's high end and highly complex load testing tool. Any change in the browser-based scripts **are not applied to the HTTP-based scripts** – which means that to perform both functional testing and load testing, **two sets of scripts must be maintained in parallel**. In addition, the HTTP-based script requires **heavy programming** to handle session identifiers, variable data, etc. The HTTP-based scripts do not perform any automatic content validation (checkpoints created in the browser-based script do not apply…) and require that the tester **manually write the code to parse the HTML string** and look for errors in the content of the page.

e-Tester implements a real development environment. Visual Basic for Applications (VBA) is **fully integrated into e-Tester** and offers all the features of the most-sold professional development environment on the market, with properties inspector, color-coded script editor, a powerful debugger, etc. This VBA environment can be used for anything that requires the user to go beyond the power of Visual Scripting: conditional branching, loops, etc. VBA also includes the ability to access the Document Object Model and make calls to a database for example. e-Tester features a powerful Custom Test Case Wizard that **automatically generates** all the test cases needed to verify Document Object Model properties – as a result, most of the time, **the users do not even have to write or edit any code**.

LOAD TESTING ACCURACY

LOAD TEST DEFINITION

Astra Load Test is used to perform load testing of the Web application. Astra Load Test uses the scripts that were created with the Virtual User Recorder, and lets you define how many virtual users you want to run of each script. The interface through which you define this load test is different than the one e-Load uses, but the idea is similar.

ERROR CHECKING

The error checking is one of the biggest flaws in Astra. While performing a load test, Astra only requests pages from the Web server. The software does not automatically even check that a page is returned. Astra also does not verify HTTP return codes, WININET error codes, or even checkpoints built into the script.

If the Web server becomes suddenly unavailable (for example if the server crashes or the network connection becomes unavailable), Astra will not show failures. **It accepts browser-displayed error pages as valid responses** from the Web application, and continues to the next page – **event though the links or buttons on which to click to navigate to the next page do not exist**! If the Web server is still available but returns the wrong page, the checkpoints in the script are not verified and Astra will not detect any error. If an error page is returned, Astra will continue to attempt navigating to the subsequent page.

On the other hand, e-Load real-time error checking features validate the structure and content of every Web page that is returned by the server. If the wrong page is returned, or a page that contains an error message is returned**, e-Load will identify this error and consider the iteration to have failed**. Of course, the load test will not be interrupted and the Virtual User will start again at the beginning of the next iteration (unless the tester elects to abort a virtual user or even stop the entire load test when an error occurs).

SCALABILITY AND REPORTING

SCALABILITY

As explained above, in the "Recording Technologies" paragraph, Astra produces two scripts: a browser-based script and a HTTP-based script. Depending on the script used, **the scalability of the tool varies dramatically**.

When using browser-based scripts, Astra relies heavily on the browser for performing its load testing. Mercury defines this mode as "*The Browser mode, which is the **default load testing mode**, and uses a real browser to emulate the load and exactly represents real-life users*." As stated, each virtual user requires a full browser to be instantiated on the load generation machine. As a result, **the scalability of Astra is poor** in this mode – the limitation stems from the CPU and the memory consumption, but also is based on the instability created by having a high number of instances of the browser running on the same PC. In their recommendations for hardware configuration, Mercury indicates that:

- A Pentium II 233 MHz supports 20 to 50 Vusers (virtual users) in browser mode
- A Pentium II/III 400 MHz supports 100 to 200 Vusers (virtual users) in browser mode

They don't provide any recommendation for a more powerful machine since **Astra cannot handle more than 200 virtual users**, even on a very powerful machine, when playing browser-based scripts (the ones that are easy to use).

Since this is the default mode, **this is also the mode that is always used during evaluations, proof-of-concepts**, etc.

But when testers want to run high-volume load tests with Astra, they are forced to use the HTTP-based scripts – the ones that are **very complex to use and require intensive programming**. Mercury defines this mode as "*The HTTP mode makes the calls to the servers with HTTP protocol which increases testing scalability*." According to Mercury's hardware configuration requirements, a Pentium II/III 400 MHz can support 200 to 500 Vusers (virtual users). Strangely enough, **this mode is never used during evaluations and proof-of-concepts**… but customers have to use it in real life!

Conversely, **with a single Visual Script** (that does not require any advanced programming), **e-Load can simulate up to 1,000 virtual users** from a single Pentium II/III 400 MHz. If the PC is more powerful, there is no limit to the number of virtual users it can generate – a Systems Integrator recently executed a test where e-Load was running **6,800 virtual users** on a fast, multi-processor machine. e-Load uses a distributed controller/agent

architecture to increase the number of virtual users that can be generated by using several machines. In addition, e-Load's breakthrough technology emulates **the exact same behavior as a browser** to represent **real-life users** – and it checks permanently the structure of the pages returned by the server to **detect errors automatically** (which Astra does not do in HTTP mode – refer to their definitions of the browser and HTTP mode above).

In addition, e-Load supports agents running on Unix machines, which lets companies that own powerful Unix servers use them to generate the load, with the same ease-of-use.

REPORTING

While the load test runs, Astra can only collect performance statistics from a limited number of Web servers. If you want to collect advanced statistics from specific application servers, you need to buy extra modules from Mercury Interactive – and if you switch from one application server to another you need to go back and spend extra to buy the specific modules that collect statistics from the new application server.

On the other hand, e-Load's built-in performance statistics collection module can collect advanced statistics from an **unlimited number of Web servers, application servers, database servers, operating systems, networks**, etc. **All the data sources are included in the standard package**.

When it comes to reporting on the results of the load test, e-TEST suite includes a **Web-based reporting module**, WebReporter, that lets anyone on the corporate network (with the appropriate set of permissions) generate their own custom reports. WebReporter includes exactly the same features as the GUI-based reporting tool. This allows engineers and systems administrators to looks at the statistics that really matter for them, and relieves the QA from having to generate and distribute all kinds of reports to everyone who has a role to play in bottleneck optimization.

CONCLUSION

Even if Astra has a nice look and feel, it is definitely **not a viable solution** for testing real-life mission critical Web applications. It is rather an **entry product** designed by Mercury to lure their customers with its ease-of-use, and then force them into using their complex LoadRunner product.

Every company which is seriously evaluating Astra is strongly encouraged **to try first to use the HTTP mode** – the only one that can be used for testing real-life applications, and which is clearly as complex to use as LoadRunner! As Astra's browser mode is the default mode, it is often the only one used during evaluations, proof-of-concepts, etc. But as users start to perform actual tests on real-life applications, **they need to switch to Astra's HTTP mode** – and this requires to **give up all the ease of use**, write lots of code like in LoadRunner, etc.

In contrast, the e-TEST suite is a **proven and reliable solution**, used by a large number of companies to perform **high volume load testing** (tens of thousands of simulated users) of their Web applications.